Vulnerability Analysis of Docker Hub Official Images and Verified Images

Ruchika Malhotra
Department of Software Engineering
Delhi Technological Univeristy
New Delhi, India
ruchikamalhotra@dtu.ac.in

Anjali Bansal
Department of Software Engineering
Delhi Technological Univeristy
New Delhi, India
anjalibansal791@gmail.com

Marouane Kessentini
Department of Computer Science & Engineering
Oakland Univeristy
Rochester, MI
kessentini@oakland.edu

2023 IEEE International Conference on Service-Oriented System Engineering (SOSE) | 979-8-3503-2239-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/SOSE58276.2023.00025

***Abstract*— Container technology is gaining significant attention as compared to virtual machines due to an increase in the use of cloud computing and containers use fewer resources as compared to virtual machines. Docker is the most widely used container technology that helps in managing and running containers. Containers use images for execution that can be created with the help of a docker file or can be downloaded from various open-source repositories. Docker uses a Docker hub repository that consists of official and verified images. As containers share the host operating system, there is a need to monitor the security of the images. In this paper, we are analyzing the vulnerabilities in official and verified docker images with the help of open-source vulnerability detection tools such as anchore, aqua trivy, docker scan and jfrog xray. This paper helps in identifying which types of images are more secure based on the number of vulnerabilities and severity of vulnerabilities and whether the number of pulls and number of stars affects the number of vulnerabilities in images.**

Keywords*— *Docker, Containers, Images, Security, Vulnerabilities, Tools

I. Introduction

The complexity of maintaining and managing application services across large distributed computing environments is minimized by cloud-based infrastructure. As cloud computing is gaining a lot of attention, the use of virtual machine has also increased. Since virtual machines provide a way to run multiple operating system on a single device but virtual machines require a lot of resources from the host. This is where containers come into the picture as containers uses few resources.

Containerization is a technique which is used to bind a software system and its dependencies, binary files, libraries and all the necessary files required to run the software into one package which is called as a container. The main objective of using containers is to deploy a software system on various platform without any platform dependency issues and compatibility issues. Docker is the most popular containerization platform which reduces the deployment time of software system so that developers quickly ship the software into production phase.

Dockerhub is the publicly available repository which consist of docker images (official and verified images). Official images are published by docker team itself and verified images are published by a certified publisher. Dockerhub also provides a service through which developers can share their images in private mode (with selected group of users) or public mode (with whole community of dockerhub users). One can reuse the existing image by pulling it from dockerhub instead of building image from scratch.

As the docker images are maintained and updated by the users, Docker doesn't have any control on how frequently the images (libraries and packages of images) get updated. The images can go months and years without any updates. If any image uses outdated libraries or any outdated packages, this may introduce some vulnerabilities in the images and may harm the security of the containers.

A vulnerability refers to any flaws and weaknesses in the system due to which an attacker can gain control of the system. These vulnerabilities can occur due to two reasons: either there are some errors in code or there is some defect in the design of the system. Vulnerabilities can be of various types such as outdated packages, buffer overflow, SQL injection, security misconfiguration etc. According to OWASP, the most common security vulnerabilities are cryptographic failures, broken access control, insecure design, identification and authentication failures, outdated components, software and data integrity failures, injection etc.

There are two terms docker images and docker containers. Docker images and docker containers are almost same thing. A docker image is a file which is a combination of binary files, library files, and all the necessary dependencies required to run an application. A docker container is a read/write copy of a docker image. When we run a docker image it builds docker container.

The growing use of containers has led to an increased need to examine security related issues in containers. Containers are more tightly integrated with the operating system of the host as compared to virtual machines. Due to this there is a risk that any container can access the sensitive and malicious data from other containers or the host system itself. Therefore, there is a need to examine the security issues in containers

The objective of this study is to analyze the security vulnerabilities in official and verified images of docker hub with the help of various open-source image scanning tools. We have formulated following research questions:

- Is there any clear difference between the total number of vulnerabilities identified by various image scanning tools in official and verified docker images?
- Is there any clear difference between the type of vulnerabilities (severity of vulnerabilities) identified in

2642-6587/23/$31.00 ©2023 IEEE
DOI 10.1109/SOSE58276.2023.00025

official and verified docker images by various image scanning tools?

- How many Common Vulnerabilities and Exposures (CVEs) were published per year in National Vulnerability Database (NVD) database for selected images?

The remaining paper is structure as follows: Section 2 describes related work. Section 3 describes methodology used in this paper to find vulnerabilities in docker images and tools used to identify vulnerabilities. Section 4 describes experimental results. In section 5, we describe conclusion and future work.

II. Related Work

The need for monitoring and controlling vulnerabilities in docker images is growing as docker is gaining popularity day by day. In this section, we will discuss previous studies that have evaluated vulnerabilities in docker images.

There exists a number of container scanning tools such as AppArmor [1], Anchore [2], Microscanner [3], Clair [4], Cilium [5], JFrog Xray [6], Dagda [7], Aqua Trivy [8], Qualys [9], Snyk [10], etc. which have been used to detect the security vulnerabilities in containers. Javed and Toor [11] in their study investigated the quality of various container scanning tools based on coverage and accuracy. Efe et al. [12] performed a detailed review of existing studies to identify the most common CVEs, methods to detect vulnerabilities, methods to detect DoS attack, and how to prevent docker images from DoS attack.

Gummaraju et al. [13] analyzed security vulnerabilities in docker hub official images. They identified that there exist more than 30% of the official images which contains high severity vulnerabilities. Zerouali et al. [14] analyzed the relationship between vulnerabilities in containers, outdatedness of containers and buggy packages installed in containers. They used technical lag to check the outdatedness of containers. They identified that all studied Debian container have vulnerabilities and buggy packages.

Shu et al. [15] introduced DIVA framework to analyze security vulnerabilities in official and community docker images. They identified that many images are outdated and there exist more than 180 security vulnerabilities on average in docker images. Zerouali et al. [16] analyzed that the vulnerabilities in javascript packages might have negative impact on docker hub images. Zerouali et al. [17] presented ConPan, an automated tool to monitor and analyze outdatedness, vulnerabilities, and bugs. This tool can be used as a CLI or integrated with other processes through its python API.

Gholami et al. [18] analyzed the package updation in docker images. They shows that there is a median of 8.6 upgradation of packages and 2.1 downgrades of packages per docker image. Wenhao and Zheng [19] analyzed docker architecture and potential security risks in docker containers. Huang et al. [20] performed a detailed analysis of existing security mechanism and threats for containers. They also introduced a framework for detecting the threats in docker containers.

III. Research Methodology

This section describes the steps followed to detect the vulnerabilities in docker images, tools used and images selected for analysis. This study follows an explorative methodology with a quantitative approach to collect and analyze the data. We use container scanning tools to detect the security vulnerabilities in official and verified docker images. We collect number of vulnerabilities identified in docker images, severity of identified vulnerabilities, unique CVEs, and the year in which CVE is published using container scanning tools. After collecting this data, we analyze this data to find the differences in container scanning tools, the differences in vulnerabilities in official and verified docker images, and how many CVEs get published per year.

A. Selection of Container Scanning Tools

There exists a lot of static and dynamic tools to scan images and containers in order to find vulnerabilities in images. Javed and Toor [11] investigated different tools such as Clair, Microscanner and Anchore for vulnerabilities in OS and non OS packages in docker images. Nowadays aqua trivy is used instead of Microscanner. We follow certain criteria to select the container scanning tools. In this study, we chose those tools that are popular, free of charge, and have been used in earlier studies. Since trial versions of paid tools have limited functionality therefore, we consider free of charge tools. We use Anchore, Aqua Trivy, Docker scan, and Jfrog Xray to identify the vulnerabilities in docker images. Table 1 shows image scanning tools used and their characteristics. All the tools used in this study can investigate both OS and non-OS packages for vulnerabilities. Aqua Trivy and jFrog Xray tools have information available regarding the database used in the scanning process and from where the tool got its information related to vulnerabilities.

TABLE I. Tools Used

Tool Name	OS Packages	Non-OS Packages	Database Information
Anchore	Yes	Yes	No
Aqua Trivy	Yes	Yes	Yes
Docker Scan	Yes	Yes	No
jFrog Xray	Yes	Yes	Yes

Aqua Trivy vulnerability database collects information from NVD (National Vulnerability Database), kube hunters, and advisories from software vendors. jFrog Xray vulnerability database collects information from NVD (National Vulnerability Database), CVE (Common Vulnerabilities Exposure) database, and security advisories from software vendors and open-source communities.

B. Selection of Docker images

Docker hub consists of a mixture of official and verified images. We consider both official and verified images on docker hub. Official images are published by docker team itself and verified images are published by a certified publisher. In this study, we consider top 5 official images and top 5 verified images for each selected official image from docker hub. Images are selected based on number of pulls, updation time period and number of stars. We choose ubuntu, nginx, mongo, mysql, and postgres official docker images and their corresponding top 5 verified images from docker hub. Table 2 shows selected official images and verified images, their number of downloads, number of stars, and latest updation time. Based on the analysis of table 2, we can say

that official images are most downloaded and most frequently used images as compared to verified images.

TABLE II. DOCKER IMAGES (OFFICIAL AND VERIFIED IMAGES)

Name of the Image	Number of Pulls	Number of Stars	Updation Time
Postgres	1B+	10K+	15 days ago
bitnami/postgresql	1B+	196	5 days ago
circleci/postgres	100M+	31	A year ago
bitnami/postgres-exporter	50M+	9	4 days ago
rapidfort/postgresql	100K+	16	5 days ago
bitnami/postgresql-repmgr	10M+	20	5 days ago
Ubuntu	1B+	10K+	6 days ago
ubuntu/apache2	1M+	58	6 days ago
ubuntu/squid	1M+	56	6 days ago
ubuntu/cortex	1M+	3	3 months ago
ubuntu/redis	100K+	18	6 months ago
ubuntu/prometheus	100K+	40	2 months ago
Mongo	1B+	9.6K	8 hours ago
bitnami/mongodb	1B+	216	4 days ago
circleci/mongo	10M+	13	A year ago
bitnami/mongodb-exporter	50M+	10	4 days ago
rapidfort/mongodb	50K+	15	A month ago
percona/percona-server-mongodb	10M+	37	A month ago
Mysql	1B+	10K+	9 days ago
circleci/mysql	100M+	29	A days ago
bitnami/mysql	100M+	86	4 days ago
bitnami/mysqld-exporter	10M+	5	4 days ago
rapidfort/mysql8-ib	50K+	0	4 days ago
rapidfort/mysql	50K+	14	10 days ago
Nginx	1B+	10K+	14 days ago
bitnami/nginx	100M+	159	6 days ago
bitnami/nginx-ingress-controller	10M+	25	6 days ago
kasmweb/nginx	1M+	6	4 days ago
rapidfort/nginx	50K+	3	6 days ago
linuxserver/nginx	50M+	193	7 days ago

We use the following steps to detect the vulnerabilities in docker images:

- Download and install docker desktop
- Download the selected images using docker pull command.

  docker pull <image name>
- Download and install the selected tools
- Start the scan of the docker image
- Save the results such as image name, tool name, total number of vulnerabilities identified, and severity wise vulnerabilities.
- Extract the CVE publication year in order to find number of CVEs published per year.

IV. RESULTS AND DISCUSSION

In this section, we describe the results collected after each scan of the images using container scanning tools. In this study, we compare the tools performance based on total number of vulnerabilities identified in docker images and severity of vulnerabilities. Table 3 and fig 1 shows the total number of vulnerabilities reported by each tool in selected official and verified images. According to table 3, there is a major difference between the total number of vulnerabilities detected by each tool. We can analyze that in most of the docker images, aqua trivy detects highest number of vulnerabilities and jFrog Xray detects least number of vulnerabilities. Based on the analysis of table 3 and fig 1, we address first research question:

RQ1: Is there any clear difference between the total number of vulnerabilities identified by various image scanning tools in official and verified docker images?

Aqua Trivy identified the highest total number of vulnerabilities (2300), anchore identified 2227 vulnerabilities, jfrog xray identified 1332 vulnerabilities, and docker scan identified the lowest total number of vulnerabilities (1213) in selected docker official and verified images.

TABLE III. TOTAL NUMBER OF VULNERABILITIES REPORTED BY EACH TOOL

Name of the Image	Anchore	Aqua Trivy	Docker Scan	jFrog Xray
Postgres	112	116	52	50
bitnami/postgresql	101	104	52	32
circleci/postgres	208	211	92	82
bitnami/postgres-exporter	81	86	45	18
rapidfort/postgresql	34	34	27	5
bitnami/postgresql-repmgr	101	104	52	32
Ubuntu	18	18	10	3
ubuntu/apache2	0	12	9	1
ubuntu/squid	0	14	10	3
ubuntu/cortex	77	63	14	57
ubuntu/redis	20	20	11	3
ubuntu/prometheus	74	82	38	43
Mongo	82	86	22	69
bitnami/mongodb	89	101	53	50
circleci/mongo	212	216	83	97
bitnami/mongodb-exporter	83	83	45	17
rapidfort/mongodb	42	53	37	25
percona/percona-server-mongodb	20	23	34	290
Mysql	7	11	21	118
circleci/mysql	231	236	111	64
bitnami/mysql	80	81	43	31
bitnami/mysqld-exporter	83	88	45	36
rapidfort/mysql8-ib	17	17	17	20
rapidfort/mysql	21	21	21	4
Nginx	140	144	102	22
bitnami/nginx	80	77	42	31
bitnami/nginx-ingress-controller	99	89	52	16
kasmweb/nginx	79	76	46	105
rapidfort/nginx	35	34	27	4
linuxserver/nginx	1	0	0	4

All the tools categorize the security vulnerabilities into four types based on severity levels: Critical, High, Medium, and Low. Table 4 and fig 2 shows the severity wise vulnerabilities in all the selected docker images. According to table 4, some of the images have zero critical vulnerabilities using all the selected tools and most of the images have higher low severity vulnerabilities. On an average, official images have less vulnerabilities than vulnerabilities in verified images. So, we can say that official images are more secure as compared to their corresponding verified images From fig 2,

we can say that linuxserver/nginx is more secure as compared to all the selected images as it has total 5 vulnerabilities using all the selected tools. Based on the analysis of table 4 and fig 2, we address second research question:

RQ2: Is there any clear difference between the type of vulnerabilities (severity of vulnerabilities) identified in official and verified docker images by various image scanning tools?

Most of the official images have high number of low severity vulnerabilities. In most of the cases, official images have least number of critical, high, and medium severity vulnerabilities as compared to verified images. So, we can conclude that official images are more secure as compared to verified images.

TABLE IV. SEVERITY WISE VULNERABILITIES REPORTED IN EACH IMAGE

Name of the Image	Critical	High	Medium	Low
Postgres	4	60	27	239
bitnami/postgresql	2	35	20	232
circleci/postgres	68	162	118	245
bitnami/postgres-exporter	2	30	20	178
rapidfort/postgresql	0	5	6	89
bitnami/postgresql-repmgr	2	35	20	232
Ubuntu	0	0	16	42
ubuntu/apache2	0	0	4	9
ubuntu/squid	0	0	7	20
ubuntu/cortex	3	101	76	31
ubuntu/redis	0	0	14	40
ubuntu/prometheus	0	41	112	84
Mongo	2	46	139	72
bitnami/mongodb	5	65	24	199
circleci/mongo	4	71	338	195
bitnami/mongodb-exporter	2	28	20	178
rapidfort/mongodb	3	40	58	106
percona/percona-server-mongodb	6	87	175	99
Mysql	2	44	74	37
circleci/mysql	61	155	94	332
bitnami/mysql	2	35	19	179
bitnami/mysqld-exporter	2	45	26	179
rapidfort/mysql8-ib	0	0	23	48
rapidfort/mysql	0	5	2	60
Nginx	8	40	89	271
bitnami/nginx	2	35	19	174
bitnami/nginx-ingress-controller	4	32	20	200
kasmweb/nginx	31	176	90	9
rapidfort/nginx	0	5	2	93
linuxserver/nginx	0	3	1	1

During scanning of the docker images, we identify that there are some vulnerabilities which occur most of the times in all the images. CVE-2022-1271, CVE-2005-2541, CVE-2022-0563, CVE-2019-9192, CVE-2017-11164, CVE-2022-29458, and CVE-2019-8457 are some of the most identified vulnerabilities in all the images using all the selected tools. We identify total 342 unique vulnerabilities in all the images. Fig 3 depicts the number of CVE ID published per year. Based on fig 3, we can answer research question 3:

- RQ3: How many Common Vulnerabilities and Exposures (CVEs) were published per year in National Vulnerability Database (NVD) database for selected images?

We identify total 342 unique vulnerabilities in all the selected images. Although most of the vulnerabilities identified are from 2022(155 total unique vulnerability), the oldest vulnerability identified is from 2005. Also, 14 vulnerabilities are published in 2023 till now.

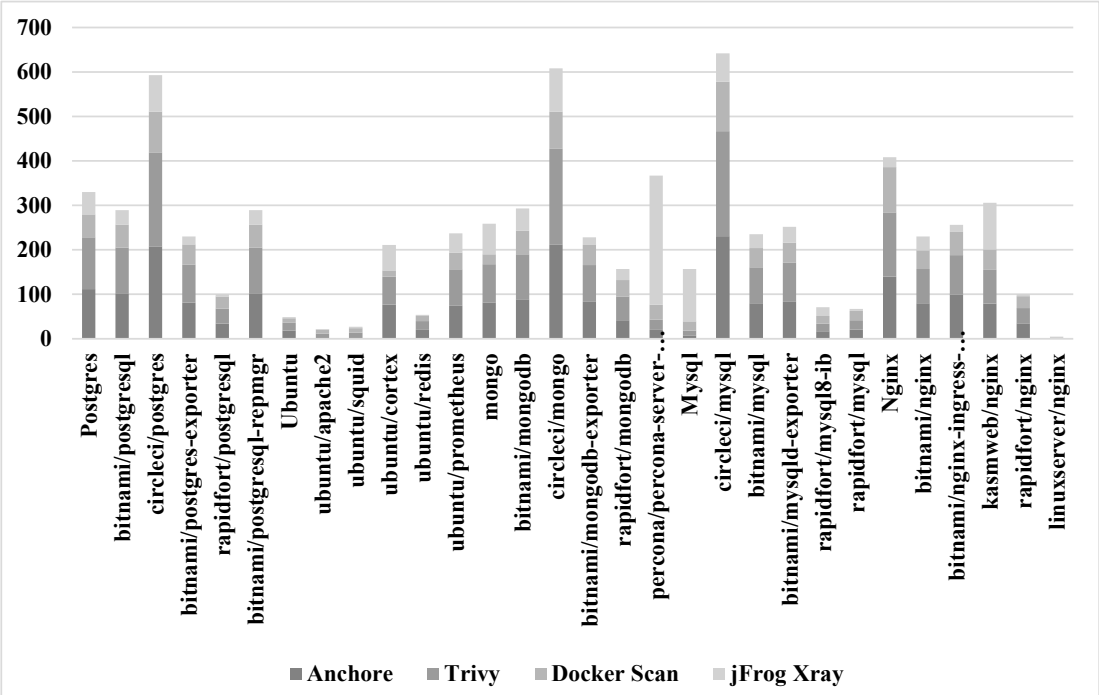

Fig. 1. Total Number of Vulnerabilities Reported by Each Tool

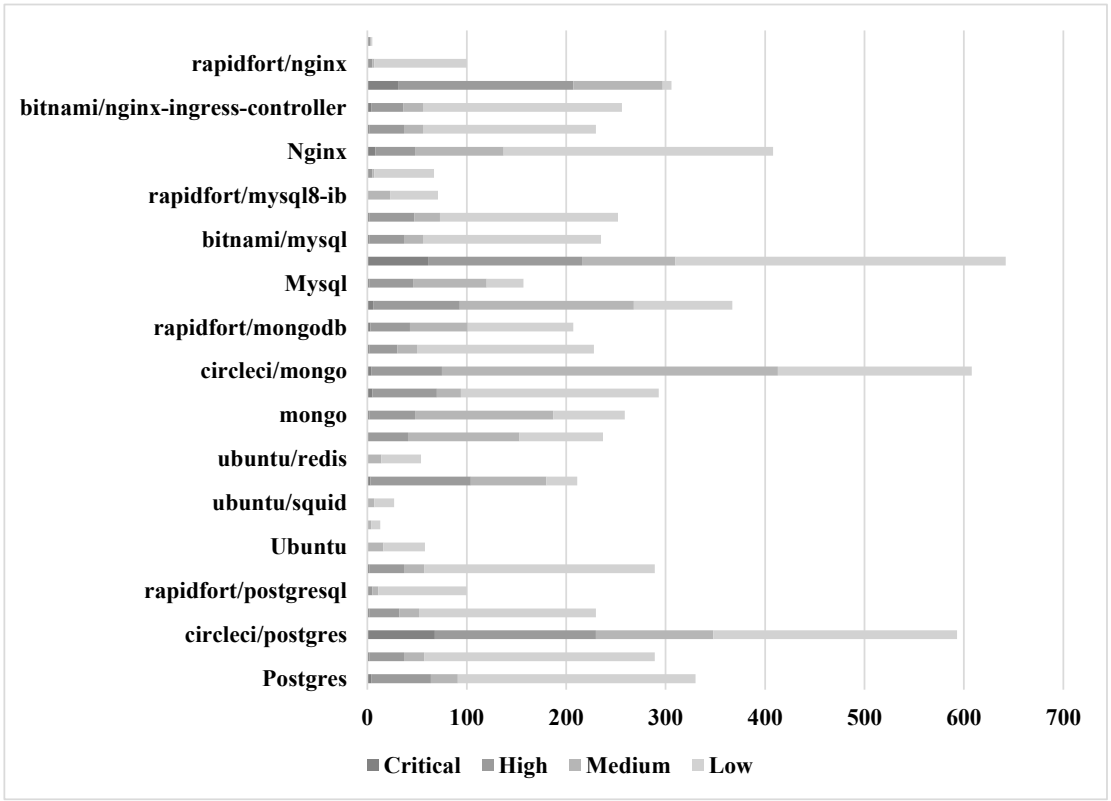

Fig. 2. Severity Wise Vulnerabilities Reported in Each Image

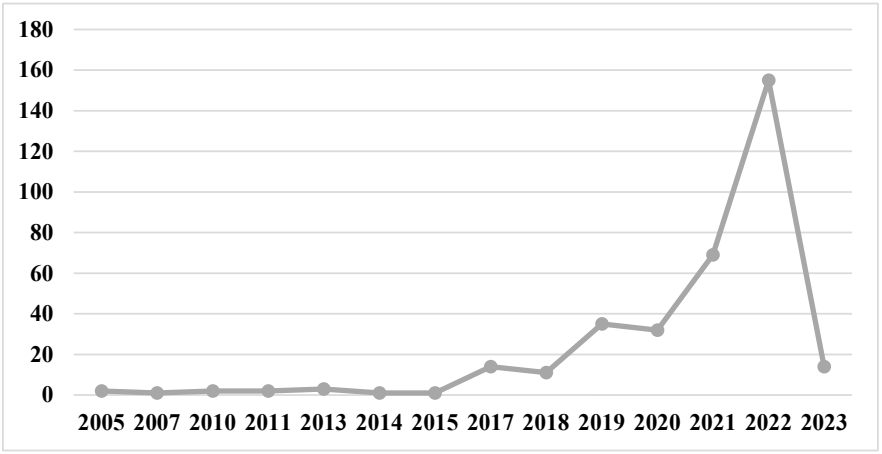

Fig. 3. CVEs Published Per Year

V. Conclusion And Future Work

This study analyze the security vulnerabilities in 5 official images and their corresponding top 5 verified images. Official and verified docker images from docker hub are selected based on number of pulls and number of stars. We use anchore, docker scan, aqua trivy, and jfrog xray to detect vulnerabilities in docker images. We analyze that number of pulls and number of stars does not affect the vulnerabilities in docker images. We find that trivy detects the higher number of vulnerabilities and docker scan detects the lowest number of vulnerabilities. We also find that official images are more secure as compared to verified images. We also analyze that same CVE ID is detected with four different severity levels: critical, high, medium and low. During the scanning of the images, we identify that some of the vulnerabilities occur most of the times in all the images. We may conclude that official images are more secure than verified images and official images are most frequently used images as compared to verified images.

As there is a major difference between the total number of vulnerabilities detected by each tool, there is a need to explore why there is a big difference in the results of the tools. This signifies the accuracy of the tools. The different severity levels reported for the same vulnerabilities also require further research.

References

[1] "Documentation · Wiki · AppArmor / apparmor · GitLab." https://gitlab.com/apparmor/apparmor/-/wikis/Documentation.

[2] "Container Vulnerability Scanning • Anchore." https://anchore.com/container-vulnerability-scanning.

[3] "MicroScanner: New Free Image Vulnerability Scanner for Developers - Aqua." https://www.aquasec.com/news/microscanner-new-free-image-vulnerability-scanner-for-developers/.

[4] "Docker Image/Container Security Scan with Clair — Installation | by Kinjal Rathod | System Weakness." https://systemweakness.com/docker-image-container-security-scan-with-clair-installation-355f80201ef5.

[5] "Cilium - Linux Native, API-Aware Networking and Security for Containers." https://cilium.io/.

[6] "Software Composition Analysis Tool - JFrog Xray." https://jfrog.com/xray/.

[7] "GitHub - eliasgranderubio/dagda: a tool to perform static analysis of known vulnerabilities, trojans, viruses, malware & other malicious threats in docker images/containers and to monitor the docker daemon and running docker containers for detecting anomalous activities." https://github.com/eliasgranderubio/dagda/.

[8] "Trivy Open Source Vulnerability Scanner | Aqua." https://www.aquasec.com/products/trivy/.

[9] "Container Security | Qualys." https://www.qualys.com/apps/container-security/.

[10] "What is container security? | Container Image Security | Snyk." https://snyk.io/learn/container-security/.

[11] O. Javed and S. Toor, "Understanding the Quality of Container Security Vulnerability Detection Tools." arXiv, 2021. doi: 10.48550/ARXIV.2101.03844.

[12] E. F. E. Ahmet, U. Aslan, and A. M. Kara, "Securing Vulnerabilities in Docker Images," *International Journal of Innovative Engineering Applications*, vol. 4, no. 1, pp. 31–39, 2020.

[13] J. Gummaraju, T. Desikan, and Y. Turner, "Over 30% of official images in docker hub contain high priority security vulnerabilities," *Technical Report*, 2015.

[14] A. Zerouali, T. Mens, G. Robles, and J. M. Gonzalez-Barahona, "On the relation between outdated docker containers, severity vulnerabilities, and bugs," in *2019 ieee 26th international conference on software analysis, evolution and reengineering (saner)*, 2019, pp. 491–501.

[15] R. Shu, X. Gu, and W. Enck, "A Study of Security Vulnerabilities on Docker Hub," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, 2017, pp. 269–280. doi: 10.1145/3029806.3029832.

[16] A. Zerouali, V. Cosentino, T. Mens, G. Robles, and J. M. Gonzalez-Barahona, "On the Impact of Outdated and Vulnerable Javascript Packages in Docker Images," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2019, pp. 619–623. doi: 10.1109/SANER.2019.8667984.

[17] A. Zerouali, V. Cosentino, G. Robles, J. M. Gonzalez-Barahona, and T. Mens, "ConPan: A Tool to Analyze Packages in Software Containers," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019, pp. 592–596. doi: 10.1109/MSR.2019.00089.

[18] S. Gholami, H. Khazaei, and C.-P. Bezemer, "Should you upgrade official docker hub images in production environments?," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, 2021, pp. 101–105.

[19] J. Wenhao and L. Zheng, "Vulnerability analysis and security research of docker container," in *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, 2020, pp. 354–357.

[20] D. Huang, H. Cui, S. Wen, and C. Huang, "Security Analysis and Threats Detection Techniques on Docker Container," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, 2019, pp. 1214–1220. doi: 10.1109/ICCC47050.2019.9064441.